THE CLAIMS

1. A method of using a virtual machine monitor and an operating system on computer hardware, the method comprising interposing the virtual machine monitor between the computer hardware and the operating system at runtime.

2. The method of claim 1, further comprising booting the operating system on the hardware before interposing the virtual machine monitor at runtime.

3. The method of claim 1, further comprising booting the virtual machine monitor on the hardware, booting the operating system on the virtual machine monitor, and devirtualizing the hardware before interposing the virtual machine monitor at runtime.

4. The method of claim 1, further comprising devirtualizing the hardware after the virtual machine monitor has been interposed.

5. The method of claim 1, wherein the computer hardware includes a CPU; and wherein the virtual machine monitor is interposed on the CPU.

6. The method of claim 5, wherein the computer hardware further includes memory, and the virtual machine monitor and the operating system each include CPU interrupt handlers; and wherein interposing the virtual machine monitor on the CPU includes causing privileged instructions to trap to the virtual machine monitor, and redirecting interrupts from the

operating system handlers to the corresponding virtual machine monitor interrupt handlers.

7. The method of claim 6, wherein the privileged instructions are caused to trap to the virtual machine monitor by causing the operating system to run at a reduced privilege level; and wherein interposing the virtual machine monitor on the CPU further includes returning control to the operating system at the reduced privilege level.

8. The method of claim 6, wherein the privileged instructions are caused to trap to the virtual machine monitor by using a kernel module of the operating system to reduce the privilege level of the operating system.

9. The method of claim 6, wherein interposing the virtual machine monitor on the CPU further includes disabling physical memory access by the operating system

10. The method of claim 6, wherein the computer hardware includes memory; and wherein interposing the virtual machine monitor on the CPU further includes loading the virtual machine monitor into the memory.

11. The method of claim 10, wherein a kernel module of the operating system is used to allocate memory within the operating system, pin the allocated memory, and load the virtual machine monitor into the pinned memory.

12. The method of claim 5, wherein the computer hardware includes memory; and wherein the virtual machine monitor is also interposed on the memory.

13. The method of claim 12, wherein interposing the virtual machine monitor on the memory includes partitioning the memory, and giving the virtual machine monitor access to at least one of the partitions.

14. The method of claim 12, wherein interposing the virtual machine monitor on the memory includes using a kernel module of the operating system to allocate a block of the memory, pin the block to prevent the operating system from using the block, and allocate the pinned block to the virtual machine monitor.

15. The method of claim 12, wherein interposing the virtual machine monitor on the memory includes constructing an *Identity* mapping of physical to machine memory; and commencing using the virtual machine monitor at runtime to manage memory translation.

16. The method of claim 5, wherein the computer hardware includes an I/O device, and wherein the virtual machine monitor is also interposed on the I/O device.

17. The method of claim 16, wherein the operating system includes dual-mode drivers that perform direct hardware control in a first mode and communicate with device drivers of the virtual machine monitor in a second mode; and wherein interposing the virtual machine monitor on the I/O device includes setting the dual-mode driver to the second mode; and

redirecting I/O interrupts from interrupt handlers in the operating system to interrupt handlers in the virtual machine monitor.

18. The method of claim 16, wherein interposing the virtual machine monitor on the I/O device includes commencing I/O emulation of the device at runtime.

19. A method of using a virtual machine monitor and an operating system on virtualized computer hardware, the method comprising devirtualizing the hardware at runtime.

20. The method of claim 19, wherein the computer hardware includes a CPU; and wherein the CPU is devirtualized at runtime.

21. The method of claim 20, wherein the computer hardware further includes physical memory, and the virtual machine monitor and the operating system each include CPU interrupt handlers; and wherein devirtualizing the CPU includes redirecting interrupts from the virtual machine monitor handlers to the corresponding operating system interrupt handlers.

22. The method of claim 21, wherein devirtualizing the CPU further includes restoring privilege level of the operating system.

23. The method of claim 21, wherein devirtualizing the CPU further includes enabling physical memory access by the operating system

24. The method of claim 21, wherein devirtualizing the CPU further includes unloading the virtual machine monitor from the memory.

25. The method of claim 19, wherein the computer hardware includes memory; and wherein the memory is devirtualized at runtime.

26. The method of claim 25, wherein memory was allocated from the operating system to the virtual machine monitor during virtualization of the memory; and wherein devirtualizing the memory includes returning the allocated memory to the operating system.

27. The method of claim 25, wherein devirtualizing the memory includes remapping physical memory so a physical-to-machine mapping becomes an *Identity* mapping; and using the operating system to manage address translation with respect to the devirtualized memory.

28. The method of claim 19, wherein the computer hardware includes an I/O device, and wherein the I/O device is devirtualized at runtime.

29. The method of claim 28, wherein the operating system includes dual-mode drivers that perform direct hardware control in a first mode and communicate with device drivers of the virtual machine monitor in a second mode; and wherein devirtualizing the I/O device includes setting the dual-mode drivers to the first mode; and redirecting I/O interrupts from handlers in the virtual machine monitor to handlers in the operating system.

30. The method of claim 28, wherein devirtualizing the I/O device includes ceasing emulation of the I/O device at runtime.

31. A computer comprising hardware, the hardware including memory, the memory encoded with an operating system, a virtual machine monitor, and means for interposing the virtual machine monitor on the hardware at runtime.

32. The computer of claim 31, wherein the hardware further includes a CPU, and the virtual machine monitor and the operating system each include CPU interrupt handlers; and wherein the interposing means causes privileged instructions to trap to the virtual machine monitor, and redirects interrupts and traps from the operating system handlers to the corresponding virtual machine monitor interrupt handlers, whereby the virtual machine monitor is interposed on the CPU at runtime.

33. The computer of claim 32, wherein the interposing means causes privileged instructions to trap to the virtual machine monitor by causing the operating system to run at a reduced privilege level; and wherein the interposing means reduces privilege level of the operating system after redirecting the interrupts, and returns control to the operating system at the reduced privilege level.

34. The computer of claim 32, wherein the interposing means includes a kernel module of the operating system for reducing the privilege level of the operating system, whereby the privileged instructions trap to the virtual machine monitor.

35. The computer of claim 32, wherein the interposing means disables physical memory access by the operating system.

36. The computer of claim 31, wherein the interposing means includes a kernel module of the operating system for allocating a block of the memory, pinning the block to prevent the operating system from using the block, and allocating the pinned block to the virtual machine monitor, whereby the virtual machine monitor is interposed on the memory at runtime.

37. The computer claim 31, wherein the interposing means constructs an *Identity* mapping of physical to machine memory; and commences using the virtual machine monitor at runtime to manage memory translation, whereby the virtual machine monitor is interposed on the memory at runtime.

38. The computer of claim 31, wherein the hardware further includes an I/O device; and wherein the interposing means includes operating system dual-mode drivers that perform direct hardware control in a first mode and communicate with device drivers of the virtual machine monitor in a second mode; and wherein the interposing means sets the dual-mode driver to the second mode; and redirects I/O interrupts from interrupt handlers in the operating system to interrupt handlers in the virtual machine monitor, whereby the virtual machine monitor is interposed on the I/O device at runtime.

39. The computer of claim 31, wherein the hardware further includes an I/O device; and wherein the operating system includes dual-mode drivers that perform direct hardware control in a first mode and communicate with device drivers of the virtual machine monitor in a second mode; and wherein the interposing means sets the dual-mode driver to the second mode; and redirects I/O interrupts from interrupt handlers in the operating

system to interrupt handlers in the virtual machine monitor, whereby the virtual machine monitor is interposed on the I/O device.

40. The computer of claim 31, wherein the hardware further includes an I/O device; and wherein the interposing means commences I/O emulation of the I/O device at runtime, whereby the virtual machine monitor is interposed on the I/O device at runtime.

41. A computer comprising hardware, the hardware including memory, the memory encoded with means for virtualizing the hardware, and means for devirtualizing the hardware at runtime.

42. The computer of claim 41, wherein the hardware further includes a CPU; and wherein the devirtualizing means devirtualizes the CPU at runtime.

43. The computer of claim 42, wherein the memory is further encoded with an operating system including interrupt handlers; wherein the virtualizing means includes interrupt handlers; and wherein the devirtualizing means redirects interrupts from the interrupt handlers of the virtualizing means to the corresponding interrupt handlers of the operating system.

44. The computer of claim 43, wherein the devirtualizing means restores privilege level of the operating system.

45. The computer of claim 43, wherein the devirtualizing means enables physical memory access by the operating system.

46. The computer of claim 41, wherein the devirtualizing means devirtualizes the memory at runtime.

47. The computer of claim 46, wherein the virtualizing means allocates memory from an operating system to the virtualizing means; and wherein the devirtualizing means returns the allocated memory to the operating system.

48. The computer of claim 46, wherein devirtualizing means remaps physical memory so a physical-to-machine mapping becomes an *Identity* mapping; and uses an operating system to manage address translation with respect to the devirtualized memory.

49. The computer of claim 41, wherein the computer hardware includes an I/O device, wherein the virtualizing means virtualizes the I/O device; and wherein the devirtualizing means devirtualizes the I/O device at runtime.

50. The computer of claim 49, wherein the memory is further encoded with an operating system including dual-mode drivers that perform direct hardware control in a first mode and communicate with device drivers of the virtualizing means in a second mode; and wherein the devirtualizing means sets the dual-mode drivers to the first mode; and redirects I/O interrupts from handlers in the virtualizing means to handlers in the operating system.

51. The computer of claim 49, wherein the devirtualizing means ceases emulation of the I/O device at runtime.

52. An article for use with an operating system on computer hardware, the article comprising software for virtualizing at least some of the hardware at runtime.

53. The article of claim 52, wherein the hardware further includes a CPU, and the virtual machine monitor and the operating system each include CPU interrupt handlers; and wherein the software causes privileged instructions to trap to the virtual machine monitor, and causes interrupts and traps to be redirected from the operating system handlers to the corresponding virtual machine monitor interrupt handlers.

54. The article of claim 53, wherein the software causes the privileged instructions to trap by reducing privilege level of the operating system, and wherein the software causes control to be returned to the operating system at the reduced privilege level.

55. The article of claim 53, wherein the software causes physical memory access by the operating system to be disabled.

56. The article of claim 52, wherein the software includes a virtual machine monitor for causing a kernel module of the operating system to allocate a block of the memory, pin the block to prevent the operating system from using the block, and allocate the pinned block to the virtual machine monitor.

57. The article of claim 52, wherein the software includes a virtual machine monitor that causes an *Identity* mapping of physical to machine memory to be constructed; and that manages memory translation at runtime.

58. The article of claim 52, wherein the hardware further includes an I/O device; and wherein the software includes an operating system dual-mode driver that performs direct hardware control in a first mode and communicates with a corresponding device driver of a virtual machine monitor in a second mode; and wherein the dual-mode driver is set to the second mode during runtime interposition; and wherein I/O interrupts are redirected from interrupt handlers in the operating system to interrupt handlers in the virtual machine monitor.

59. The article of claim 52, wherein the hardware further includes an I/O device; and wherein the operating system includes dual-mode drivers that perform direct hardware control in a first mode and communicate with device drivers of the virtual machine monitor in a second mode; and wherein the dual-mode driver are set to the second mode during interposition; and wherein I/O interrupts are redirected from interrupt handlers in the operating system to interrupt handlers in the virtual machine monitor.

60. The article of claim 52, wherein the hardware further includes an I/O device; and wherein the software causes I/O emulation of the I/O device to commence at runtime.

61. An article for running an operating system and a virtual machine monitor on a computer, the computer including an I/O device, the article comprising computer memory encoded with an I/O driver having first and second modes of operation, the I/O driver operable in the first mode to interface directly between the operating system and the I/O device, the I/O driver operable in the second mode to interface between the operating system and a corresponding I/O driver of the virtual machine monitor.

62. An article for use with an operating system on computer hardware, the article comprising software for devirtualizing at least some virtualized hardware at runtime.

63. The article of claim 62, wherein the hardware further includes a CPU; and wherein the software causes the CPU to be devirtualized at runtime.

64. The article of claim 63, wherein the memory is further encoded with an operating system including first interrupt handlers; wherein the software includes second interrupt handlers; and wherein the software causes interrupts to be redirected from the second interrupt handlers to the corresponding first interrupt handlers.

65. The article of claim 64, wherein the software causes privilege level of the operating system to be restored.

66. The article of claim 64, wherein the software causes physical memory access by the operating system to be enabled.

67. The article of claim 62, wherein the software causes the memory to be devirtualized at runtime.

68. The article of claim 67, wherein if a part of the memory was allocated from an operating system to a virtual machine monitor prior to the runtime devirtualization, the software causes the allocated memory to be returned to the operating system as part of the runtime devirtualization.

69. The article of claim 67, wherein the software causes physical memory to be remapped so a physical-to-machine mapping becomes an *Identity* mapping; and wherein the software allows an operating system to manage address translation with respect to the devirtualized memory.

70. The article of claim 62, wherein the virtualized hardware includes an I/O device; and wherein the software causes the I/O device to be devirtualized at runtime.

71. The article of claim 70, wherein the memory is further encoded with an operating system including dual-mode drivers that perform direct hardware control in a first mode and communicate with virtual device drivers in a second mode; and wherein the software causes the dual-mode drivers to be set to the first mode.

72. The article of claim 70, wherein the software causes emulation of the I/O device to cease at runtime.